The Impact of Name-Matching and Blocking on Author Disambiguation

Backes, Tobias

The Impact of Name-Matching and Blocking on Author Disambiguation

Tobias Backes
GESIS Leibniz Institute for the Social Sciences
tbackes@coli.uni-saarland.de

ABSTRACT

In this work, we address the problem of blocking in the context of author name disambiguation. We describe a framework that formalizes different ways of name-matching to determine which names could potentially refer to the same author. We focus on name variations that follow from specifying a name with different completeness (i.e. full first name or only initial). We extend this framework by a simple way to define traditional, new and custom blocking schemes. Then, we evaluate different old and new schemes in the Web of Science. In this context we define and compare a new type of blocking schemes. Based on these results, we discuss the question whether name-matching can be used in blocking evaluation as a replacement of annotated author identifiers. Finally, we argue that blocking can have a strong impact on the application and evaluation of author disambiguation.

CCS CONCEPTS

• **Information systems → Entity resolution**;

KEYWORDS

Author Disambiguation; Blocking; Name-Matching

ACM Reference Format:
Tobias Backes. 2018. The Impact of Name-Matching and Blocking on Author Disambiguation. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271699

1 INTRODUCTION

In *author name disambiguation (AND)*, a set of author names mentioned on (scientific) documents is clustered into separate persons. This resolves *author name homonymy* (different authors can share the same name). A related problem is *author name synonymy*, where the same author has different names. Descriptions of the homonymy problem and its consequences are frequently found in the literature (i.e., in [14, 19]). The synonymy problem is not as present. To deal with different names for one and the same author, methods often referred to as *blocking* are applied. Naturally, the set of author mentions is partitioned by different names. Blocking partitions this set of names into subsets (blocks) while minimizing (1) block size and (2) number of synonymous author mentions separated across blocks. During AND, each block is partitioned further by authors. Theoretically, blocking could be omitted altogether. Nothing prevents a hypothetical, super-effective AND method from discriminating perfectly among a single giant block of author mentions. However, not only would the computational complexity of AND (roughly quadratic) render this approach infeasible, but we would also have to combine the constraints of name-matching with the similarities of our disambiguation method. This could be done in a constrained clustering setup, but constrained clustering is NP-hard [2]. Therefore, blocking is usually assumed a preprocessing step to AND. Formally, we have

- X, the set of author mentions x
- $\mathcal{N}$, the set of names N, partitioning X
- $\mathcal{A}, \mathcal{G}$, the sets of system/gold authors A, G, partitioning X
- $\mathcal{B}$ (*blocking*), the set of blocks B, partitioning X, $\mathcal{N}$ and $\mathcal{A}$

Due to the interplay of synonymy and homonymy, names N and system author clusters A are on the same level in this hierarchy. System authors and names both partition the blocks B they are in, but often differently. Gold author clusters G might be split across different blocks, showing blocking errors. Unfortunately, gold authors are never fully known in real world datasets. However, two indicators might be available: (1) Author mentions share an annotated *author identifier*. For example, in the Web of Science, a subset of the author mentions have been annotated with a researcher-ID assigning them to real world authors. (2) Mentions have *matching names*. Two names match if they could refer to the same author. For example, it is reasonable to assume that *John Doe* refers to a different person than *Jack Doe*. One might think that different names refer to different persons, so that $\mathcal{B}$ could be obtained by looking at each name N separately ($\mathcal{B} = \mathcal{N}$). However, two types of factors introduce name synonymy: (a) factors like spelling mistakes, different formats, short-forms, language-specificity or name changes; (b) different completeness in the specification of a name. In this work, we confine ourselves to the latter, as it has not been addressed in previous work and constitutes a basic problem beyond error-correction and normalization. Consider *John Doe* and *J. H. Doe* which might both refer to the same person. Both names give four pieces of information: In the first case, we are given the initial of the surname, the full surname, the initial of the first name and the full first name. In the second case, we are not given the full first name, but the initial of the second name. As is often done in the context of AND, we could normalize both names to *J. Doe*, but that would also include *Jack Doe* and many other names. Apparently, the blocking problem cannot be solved by the usual normalization.

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271699

In this work, we show that name-matching and blocking have a strong influence on AND. We propose a formal framework that combines both aspects to foster further investigation into effective blocking schemes. Also, we hope to establish new standards for the evaluation of AND methods that better regard the influence of blocking. We assume that author identifiers are only available for evaluation purposes and can therefore not be used for blocking. Furthermore, we assume that the usual normalization steps have been applied and we are left with names that can be parsed into a format with eight fields of information: the initial and the full string of up to three first names and the surname. Now we can define the *MATCH*-relation over all pairs (N, N') of names, which holds if and only if each field of N has either the same value as the respective field in N' or at least one of them has no value specified. The main problem is that this relation is symmetric, reflexive, but not transitive. For example, *J. Doe* can refer to *John Doe* or *Jack Doe*, but *John Doe* cannot refer to the same person as *Jack Doe*. Therefore, the MATCH-relation cannot be used directly to partition the mention space X. It can still be used to evaluate the blocking $\mathcal{B}$, although one has to be aware that perfect results are impossible (as in the last example). This means that we strive to obtain a partitioning $\mathcal{B}$ of author names (and the mentions that carry them) in which matches and ultimately also equal identifiers are captured within single blocks and not across them (measured as Recall) while keeping the blocks relatively homogeneous (measured as Precision). We evaluate to which extend different blocking schemes are able to satisfy these objectives.

As the author ambiguity problem is solved by a combination of blocking and disambiguation, two questions arise that need to be answered in order to investigate the impact of blocking on disambiguation: (i) How much is already achieved by blocking? (ii) Which mistakes are introduced that cannot be made up for during disambiguation? The answer to these questions can be given as Precision and Recall values. The collection X as it is has perfect Recall (since all correct pairs of author mentions are included) and the lowest possible Precision (since all incorrect pairs are also included). Both blocking and disambiguation lower Recall and increase Precision. Therefore, we can specify the above questions: (i) How much Precision is already achieved by blocking (blocking Precision, bPrec)? (ii) How much Recall is already lost (blocking Recall, bRec)? Considering names as disjoint sets of mentions, we note that $1 - bPrec$ is *homonymy rate* (how many mention pairs with the same name refer to different persons?) and $1 - bRec$ is *synonymy rate* (how many co-referring pairs have different names?). The blocking scheme defines what "same name" means. The urgency of the above questions is underlined by the fact that most work on AND does not report which portion of their method's Precision is due to blocking and compute Recall only within the disambiguated block, thereby ignoring mistakes across blocks. We investigate the following research questions:

RQ1 How to structure names, their variations and relations?
RQ2 How to include name-matching in the blocking?
RQ3 How do different blocking schemes perform?
RQ4 Are there better blocking schemes?
RQ5 Can name-matching substitute author annotation?
RQ6 What impact has blocking on (the evaluation of) AND?

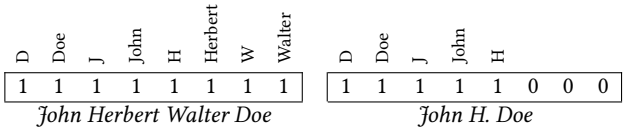

Figure 1: Two examples of author name type representations

In the next section, we discuss related work. In Section 3, we show how to represent names in a directed acyclic graph and formally define a number of relations that are helpful in this context (**RQ1**). In Section 4 we use this formalism to define different blocking schemes (**RQ2**), some of which are new (**RQ4**). In Section 5, we give implementational details. In Section 6, we evaluate different blocking schemes to answer **RQ3**, **RQ5** and also **RQ6**. The conclusion addresses all research questions, but puts particular focus on **RQ5** and **RQ6**.

2 RELATED WORK

Author Name Ambiguity. Harzing [7] compares ambiguity in names from different cultural backgrounds. She finds that the extend of ambiguity influences scientific performance indicators. This claim is supported by Strotmann and Zhao [21], who investigate the impact of AND on citation networks. They show that a number of researchers are top-ranked because they are mixed with others that have the same name. Kramer, Momeni and Mayr [13] evaluate gold author information in the Web of Science. They find that the researcher-ID is a good author identifier. Ferreiro, Goncalves and Laender [4] give an overview over different approaches to AND. They introduce a taxonomy that distinguishes among others between author grouping (resolving synonymy) and author assignment (homonymy). In a more recent comparison, Hussain and Ashger [8] discuss AND methods since 2010 and try to pinpoint strengths and weaknesses of the selected approaches. They use a different taxonomy focusing on the technology used (i.e., machine learning, graph-based, etc.).

Author Name Disambiguation. AND can be seen as a supervised or unsupervised problem. With supervision, either author mentions are assigned to existing authors, or pairs of author mentions are classified as belonging to the same author. De Carvalho et al. [3] present a heuristic method that iteratively assigns author mentions to already discovered authors or starts a new one. Although computationally practical, this setting is more difficult than unsupervised clustering. They classify a selection of names grouped by surname and first initial. Other popular supervised approaches are by Wang et al. [25], who stand out by using author affiliation history over selected names and Onodera et al. [17], who use logistic regression and include manual checking into their workflow. A sophisticated semi-supervised solution is presented by Levin et al. [15]. They use certain high-precision features like email-addresses to automatically create training data for their binary classifier. Blocking is done by surname and the first two available initials. Unsupervised approaches *cluster* author mentions into separate authors. Mostly due to the effectiveness of co- and referenced authors as features, a number of graph-based approaches have been suggested, among them Tang and Walsh [23], a purely graph-based method that is

applied in particular to the difficult case of Chinese names. Apparently they evaluate only two names. Other interesting approaches are by Tang et al. [22], who use Markov Random Fields over selected names and Gurney et al. [6], who include name information as features and focus on dealing with empty fields. The latter evaluate blocking by surname as well as surname and first initial. Schulz et al. [20], acknowledge that "size dependent biases can skew aggregate algorithm performance measures". They also suggest using name information unused in blocking to evaluate the precision of their AND method. None of the above papers report blocking Precision or Recall and it is unclear if any group computes Recall over the entire collection. Using only selected names obviously counteracts the evaluation of blocking schemes.

Blocking for Author Name Disambiguation. Most work on AND does not put any focus on the blocking scheme and usually applies very simple options without discussing alternatives. A popular scheme is to generalize all names to *surname, first initial* (as for example in [15]). In this case, only a marginal loss of Recall is expected, but blocks can become extremely large for the most frequent surnames like *Wang*, *Lee* or *Smith*. In the context of evaluating their AND method, Torvik and Smalheiser [24] actually determine Recall of this blocking scheme and report $\sim 98.8\%$ due to last name differences. We obtain similar numbers. Milojevic [16] presents a simple baseline for AND, which amounts to evaluating a blocking scheme with one *or two* initials in terms of Precision and Recall. The option of using two initials already decreases the maximal block size considerably. Unfortunately, Milojevic neither uses Recall nor a large, popular dataset for evaluation. A recent study by Kim [11], compares the distinction of authors by surname and all first name initials against more complex AND approaches. They shed a light on the impact of data selection and annotation patterns on the performance of these methods. Another important recent reference on blocking in the context of AND is by Kim, Sefid and Giles [12]. They point out that 'standard' blocking means *conjunctive* combination of boolean predicates, i.e., $surname(Doe) \wedge firstInit(J)$. Applying such 'standard' blocking implies that a block contains exactly the set of names that generalize to a common ancestor name. What happens if a piece of information is not available (i.e., no first initial given), is usually not specified. In *disjoint* blocking, blocks may overlap. According to Kim, Sefid and Giles, disjoint blocks are rather inconvenient as one name might then be assigned to different blocks. Instead of partitioning all mentions by blocks, pairs of mentions are determined for which some AND similarity is computed and a threshold is applied. The transitivity problem is not solved but postponed. Therefore, in our work, we use conjunctive predicates. Related work on name-matching focuses on normalization issues. The best reference for this topic is by Christen [1], who analyses the impact of many different sources of error in name normalization. Like Gurney et al. [6], in our work, we consider this problem already solved. Although they too focus on these normalization problems, our work has many accordances with work by Galvez and Moya-Anegón [5], who also use a graph structure and rely on the notion of blocks being equivalence classes over some relation. Instead of using a hierarchical structure as we do, they define a finite-state machine that parses a name from left to right and conflates different standards. In doing so, they basically combine normalization and name matching. However, they do not investigate the use of this technique as a blocking method for AND. Kardes et al. [10] discuss the implementation of graph-based blocking and also make use of the transitive closure to find connected components in this graph.

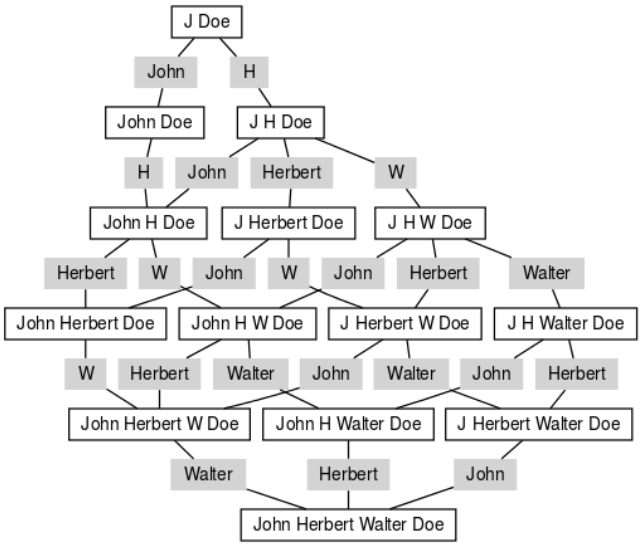

Figure 2: Part of the graph structuring all name variations

3 GRAPH STRUCTURE OF NAME MATCHES

Author names can be represented as instantiations of a number of types. Using up to three first names of an author, this results in 30 different types (see Figure 1) which could all reference the same person, but have different completeness. The most general name is the empty name and the most specific is a name like *John Herbert Walter Doe*. A more frequent case is i.e., *John H. Doe*. We assume it is not possible to specify a first name (by name or initial) without having indicated the existence of all his previous first names (by name or initial). So for example we have to assume that *John W. Doe* is a different person than *John H. W. Doe*. Given pairs (N, N') over a set of names $\mathcal{N}$, we make two important observations:

(1) Sometimes N and N' *match*
(2) Sometimes N is a *parent* of N'

We say that N, N' match, if they could both refer to the same author, because their representations are not contradictory (in each field they either share a value or the value is unspecified in at least one of the two). We say N is a parent of N' if N is a direct generalization of N', i.e., *J. H. Doe* is a direct generalization of *John H. Doe*. The MATCH-relation is the set of all pairs $(N, N') \in \mathcal{N} \times \mathcal{N}$ such that N and N' match. The PARENT relation is the set of all pairs such that N is the parent of N'. Note that we consider name matching a phenomenon, not a method. Blocking is the method that derives a partitioning from this phenomenon.

To give structure to the MATCH relation, we want to describe it with the help of the PARENT relation. The latter creates relatively descriptive graphs as shown in Figure 2. Furthermore, we want to achieve that by default, no matching names are split across blocks.

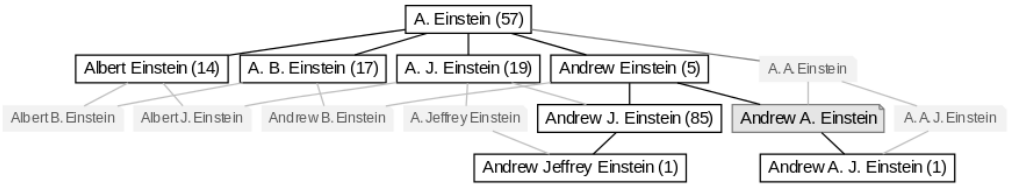

Figure 3: Reachability is fully expressed without the transparent names, which however can be relevant when blocking (i.e., isolating *A. Einstein*). The framed nodes form a subgraph reduced by minimal spanning tree to hide irrelevant nodes.

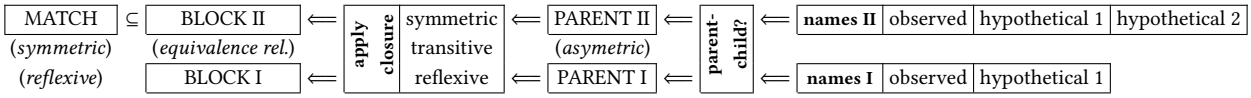

Figure 4: Using the additional set B of hypothetical/unobserved names, the MATCH-relation becomes a subset of the BLOCK-relation. This is not the case in the traditional view on author name blocks. Note that the MATCH-relation is *not* transitive.

All names that are in the PARENT relation match (so PARENT is a subset of MATCH). But not all names that match are in the PARENT relation (i.e., *John Doe* and *J. H. Doe*). However, given that the PARENT relation forms a connected graph, all names that match are in its symmetric transitive reflexive closure. We define this closure to be the BLOCK relation. Still, as discussed before, not necessarily all names in a block match (i.e., *John Doe* and *Jack Doe*). Therefore, the MATCH relation is only a subset of the BLOCK relation. Being symmetric, transitive and reflexive, the BLOCK relation is an equivalence relation and partitions $\mathcal{N}$ into equivalence classes. These coherences are displayed in Figure 4.

As seen in Figure 2, the PARENT relation can also be viewed as a directed acyclic graph over N. We note that we reach a child by either completing or adding an initial (if possible) and that its level in the tree is the number of non-empty fields in its representation. However, with the BLOCK and MATCH relation in mind, we have to add *hypothetical names* unobserved in the collection to $\mathcal{N}$ in order to ensure connectivity. For example in Figure 3, if we have seen *Andrew Einstein* and *Andrew A. J. Einstein*, we have to add *Andrew A. Einstein* to be able to link the two by the PARENT relation. Or if we have seen *A. B. Einstein* and *Andrew* Einstein, assuming both would refer to the same person, the name *Andrew B. Einstein* would be the most specific name to hypothesize for this person. It makes sense to distinguish two types of hypothetical names:

(1) Names that link observed names by being 'between' them *Andrew A. Einstein* with *Andrew Einstein* and *Andrew A. J. Einstein*
(2) Names that link observed names by being 'below' them *Andrew B. Einstein* with *A. B. Einstein* and *Andrew* Einstein

If we add only the first type of hypothetical names and apply the PARENT relation, we are left with a number of acyclic subgraphs rooted in the most general observed names. Applying the closure, we obtain one block for each of these subgraphs. However, this does not necessarily subsume the MATCH relation. For example if *J. Doe* is not observed, *J. H. Doe* and *John Doe* are not connected and therefore not in the same block even though they could refer to the same person. If we wanted to achieve this, we could of course introduce a hypothetical *J. Doe* (their *most specific generalization*), but this would also connect *Jack Doe* with the two. Adding hypothetical generalizations (that are not specifications of observed names) introduces unnecessary combinations of non-matching names and thereby lowers Precision of the resulting blocks. Nevertheless, this is what is usually done in the literature, when only generalization is applied to obtain blocks. Instead, we propose to add the *most general specification* of every pair of names that could refer to the same person. This really just models what follows from assuming the two names to be co-referring: If *J. H. Doe* and *John Doe* refer to the same author, so does *John H. Doe* (which is much less ambiguous than *J. Doe*). The BLOCK relation that we obtain from applying the closure on the PARENT relation over $\mathcal{N}$ with both types of hypothetical names added is then in fact a superset of the MATCH relation (Figure 4). This is a good starting point to apply different blocking schemes that break up the initial BLOCK relation into more convenient smaller parts (and sacrifice some matching pairs).

In order to fully represent the dataset in our name graph, we store the frequency of a name in the respective node. We say a node *covers* as many names as all of his children cover. A node covers at least as many nodes as his frequency count. When we add a name as a node to the graph, we increase the cover count of each ancestor by the number of mentions of that name such that the cover count of a name always gives the number of author mentions that it generalizes. Hypothetical nodes have a frequency of zero. To build the graph, we first include all observed names and their ancestors, which roots the graph in the empty name. This includes all hypothetical nodes of the first type as explained above. Then we add all hypothetical nodes of the second type for each pair of matching observed names. Finally, we remove all hypothetical nodes that belong neither to the first nor the second type.

4 BLOCKING IN THE NAME GRAPH

As has been briefly mentioned above, the traditional way to partition the set of names (and thereby author mentions) into blocks is to generalize all names to a fixed type (i.e., *J. H. Doe*). However, this is not necessarily satisfying as on the one hand *J. H. Doe* links many names that are clearly not co-referring (i.e., *Jack* and *John*) and on the other hand it is not clear what happens if an observed name does not generalize to the fixed type (i.e., *John Doe*). To solve the latter problem, one might generalize to all available initials (i.e., *John Doe* to *J. Doe* and *John H. Doe* to *J. H. Doe*), but this is incoherent

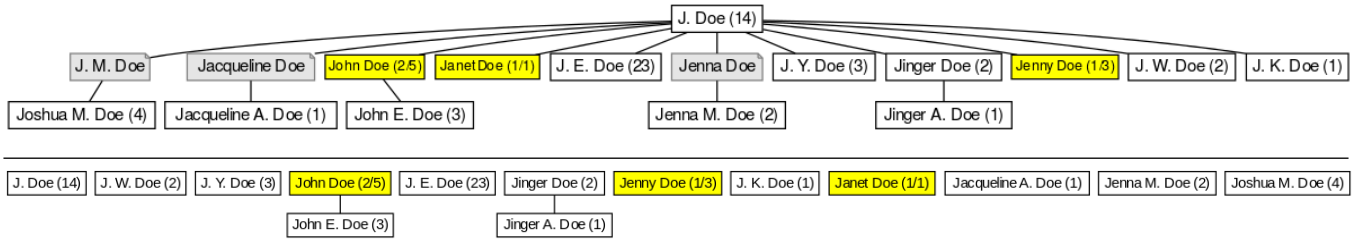

Figure 5: The initial graph consists of a single, highly connected block (here reduced). Isolating all nodes without full first name, we are left with twelve blocks – but names are separated that could actually refer to the same author. Counts are in brackets; hypothetical nodes have no counts; (2/5) means two out of five mentions annotated with rID.

in that *John H. Doe* is a specification of both types. Therefore, we propose a different formalism: We first model which names could co-refer (the MATCH relation) with the help of the PARENT relation as described above. Then we apply the closure to the PARENT relation to obtain a BLOCK relation with perfect Recall in terms of name-matching (no matching names are split across blocks). The BLOCK relation is an equivalence relation and partitions the set of author names into equivalence classes (*blocks*). So any pair of names for which there exists a connecting path in the graph has to be in the same block. If there is a *J. Doe*, a *Jack Doe* and a *John H. Doe*, then we would have to put them all in the same block to avoid a possible loss in Recall. To some degree, this is not practical: in the extreme case, one unnamed author mention (a result of anonymization) suffices to connect all names in the entire collection. Therefore, we allow blocking schemes to *isolate* names by detaching them from all parents and children. For example, we might want to isolate all names without full first name as they are simply too general. Figure 5 shows the subgraph under *J. Doe* and how it is split up by isolating this unspecific name. A blocking scheme is then simply the set S of names to isolate. This set is usually implicitly described but could also be described or extended in an explicit way to account for particularly difficult cases. Isolated names are then considered single blocks (i.e., only containing mentions with the exact name *J. Doe*). This is not particularly satisfying, but at least for popular names, there really is no alternative due to exploding computational complexity of AND.

Blocking scheme type 'field-isolate'. It can be observed that due to the transitivity of the BLOCK relation, in large collections like the Web of Science all names are somehow connected, like for example *Jack Doe*, *J. H. Doe* and *John Doe* connected by (hypothetical) nodes *Jack H. Doe* and *John H. Doe*. It is therefore practical to take the risk of sacrificing some Recall by isolating certain highly connected nodes in the graph to achieve a well-behaved partition. As explained above, blocking means isolating in the PARENT relation all names from a set S (the blocking scheme) and applying the reflexive symmetric transitive closure. We identify the following obvious blocking schemes:

- S is all names without first initial [*f2*]
- S is all names without full first name [*f3*]
- S is all names without second initial [*f4*]

Instead of completely isolating names, we can also remove specific connections. Here, we note that the PARENT relation is the union of the ADD relation (add an initial) and the COMPLETE relation (complete and initial). By subtracting the ADD relation from the BLOCK relation, we implement the following blocking scheme:

- Reduce all names to their surname and *all* initials [*inits*]

We call this type of schemes *field-isolate*. These schemes are commonly found in the literature, although it is usually not mentioned what happens with names that are in S.

Blocking scheme type 'entropy-isolate'. There are more advanced blocking schemes than fixed types. For example, Milojevic introduces a quite complicated scheme: *(a) J. Doe* generalizes to *J. Doe, (b) John Doe* generalizes to *J. Doe, (c) John H. Doe* generalizes to *J. H. Doe* ***unless*** there is no other *J. X. Doe*, in which case it generalizes to *J. Doe* and (d) anything more general than *J. Doe* is not considered. Obviously, this approach is trying to utilize the second initial if it helps discrimination among siblings and ignore it, if it only separates child and parent (which by definition could both refer to the same person). However, neither is the frequency of observation considered, nor the full first name or any other additional information in the name. Also, the constant of one sibling being enough is a rather arbitrary choice. Absorbing the basic idea, we introduce a straightforward scheme called *entropy-isolate*. Here, we simply calculate the normalized entropy H of the discrete probability distribution p_N over the children $C(N)$ of name N and determine that it should be isolated if H is above a certain threshold t:

$$H(p_N) = -\sum_{N' \in C(N)} p_N(N') \cdot \frac{\log(p_N(N'))}{\log(|C(N)|)}$$

The distribution p_N is over the *cover* counts of N's children. We isolate $S = \{N : H(p_N) > t\}$ and try the following values for t to get an overview of the scheme's behavior:

- 0 [*e0*], 0.5 [*e5*], 0.75 [*e7*], 0.875 [*e8*], 0.9375 [*e9*]

A higher threshold means accepting more competition among a name's children before we separate them by isolating their parent.

5 IMPLEMENTATIONAL DETAILS

The PARENT relation can be represented as a (sparse) boolean $|\mathcal{N}| \times |\mathcal{N}|$ matrix. We create one such matrix for each surname, such that $\mathcal{N}$ is only the set of names with the current surname. All hypothetical ancestors of all observed names (*type 1*) and the most general specification of all matching observed names (*type 2*) are also part of this matrix. We keep the information which of the indices represent hypothetical names, such that we can select a submatrix with the rest. We distinguish the matrix $\mathcal{P}$, the reflexive closure of the PARENT relation and the matrix Q, the symmetric

eval	description	matrix		suboptimal Precision	suboptimal Recall
$\mathcal{B}/\mathcal{M}$	BLOCK vs. MATCH	$\mathcal{B}$	$\mathcal{M}$	*blocks contain names that do not match*	*blocking separates matching names*
$\mathcal{B}/\mathcal{G}$	BLOCK vs. GOLD	$\mathcal{B}$	$\mathcal{G}$	*blocks contain mentions with different rIDs*	*blocking separates mentions with the same rID*
$\mathcal{M}/\mathcal{G}$	MATCH vs. GOLD	$\mathcal{M}$	$\mathcal{G}$	*two mentions name-match but have different rIDs*	*two mentions do not name-match but have same rID*

Table 1: Different setups compared in our experiments.

scheme	description
f2	*isolate all names without first initial (surname-first initial)*
f3	*isolate all names without full first name*
f4	*isolate all names without second initial*
inits	*reduce all names to surname and all initials*
e0/5/ 7/8/9	*isolate all names with entropy of count distribution over children smaller than t*

Table 2: Blocking schemes compared in our experiments.

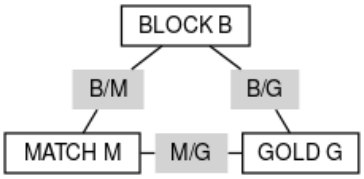

Figure 6: Evaluations $\mathcal{B}/\mathcal{G}$, $\mathcal{B}/\mathcal{M}$ and $\mathcal{M}/\mathcal{G}$.

reflexive closure. To both, we apply the blocking scheme S by setting the respective rows and columns to zero, creating matrices $\mathcal{P}'$ and $\mathcal{Q}'$. Then, we have to remove all hypothetical names that are neither 'between' nor 'below' observed names. To obtain these, we apply the transitive closure to $\mathcal{P}'$ which gives us the descendents (and ancestors) of each name. As $\mathcal{P}'$ is a directed acyclic graph, the closure can be calculated relatively efficiently [9, 18].Then, we identify all hypothetical names that (a) have no observed ancestors or (b) have no observed descendents and at most one observed parent. Then, we select the respective submatrix of $\mathcal{Q}'$ and apply the transitive closure. The complexity of this operation depends heavily on the sparsity of the result matrix, which has been increased a lot by applying the blocking scheme and deselecting all irrelevant hypothetical names. Although this is the bottleneck, it should be noted that any blocking scheme that leads to problems at this point will create much greater problems during AND. So feasibility is mostly a question of selecting an appropriate blocking scheme. The result of the transitive closure on $\mathcal{Q}'$ is the blocking matrix $\mathcal{B}$ that specifies for each pair of names under the current surname whether they are in the same block. As every name can represent multiple observed author mentions, we can multiply each cell in this matrix with the product of the observation frequencies of the respective two names to obtain the number of mention pairs in the same block with the respective names: $\mathcal{B}_{i,j} := \mathcal{B}_{i,j} \cdot |N_i| \cdot |N_j|$. This is relevant for evaluation, where we compare $\mathcal{B}$ directly against other gold standard matrices.

6 EXPERIMENTAL EVALUATION

6.1 Data

Our experiments are conducted on the Web of Science corpus with more than 150 million author mentions. Over seven million of these are annotated with researcherID's (*rID*) that allow us to evaluate the performance of different blocking schemes. Details regarding the distribution of rID's in the Web of Science are given in [13]. These ID's establish the GOLD relation which holds between annotated author mentions: either a pair of mentions references the same author (if their rID's are the same) or not. The GOLD relation is an equivalence relation and partitions the annotated portion of X into gold blocks. The amount of homonymy and synonymy in the gold data differs considerably between different surnames. For example under the surname *Alves*, 2748 different names are counted and 3933 author mentions have been assigned with an rID. Considering each name separately, the homonymy rate for this surname is only 2% and the synonymy rate is only 12%. However, under *Becker*, 2178 different names are counted and 2395 author mentions have been assigned with an rID. The homonymy rate for this surname is 35% and the synonymy rate is 36%. Alternatively, Precision and Recall of comparing name-matching directly against gold annotation ($\mathcal{M}/\mathcal{G}$, Figure 11, see below) can be used as a measure for homonymy and synonymy, respectively.

6.2 Experimental Setup

To evaluate a blocking scheme, we compare the following three matrices corresponding to the three relations introduced earlier:

(1) Blocking matrix $\mathcal{B}$ (BLOCK)
$\mathcal{B}_{i,j} = 1$ iff names i, j are in the same block
(2) Name-matching matrix $\mathcal{M}$ (MATCH)
$\mathcal{M}_{i,j} = 1$ iff names i, j could refer to the same person
(3) Gold-standard matrix $\mathcal{G}$ (GOLD)
$\mathcal{G}_{i,j} = 1$ iff mentions i, j have the same rID

We can compare all these matrices with one-another in terms of Precision and Recall (Figure 6): $\mathcal{B}$ against $\mathcal{M}$ ($\mathcal{B}/\mathcal{M}$), $\mathcal{B}$ against $\mathcal{G}$ ($\mathcal{B}/\mathcal{G}$) and $\mathcal{M}$ against $\mathcal{G}$ ($\mathcal{M}/\mathcal{G}$). We note that $\mathcal{B}$ is over all *observed names* with the current surname, whether one of the name's mentions has been assigned an rID or not. The same holds for $\mathcal{M}$. The gold-standard $\mathcal{G}$ is over *mentions*, not names, and only those that have been annotated with an rID. So for $\mathcal{B}/\mathcal{M}$, we can directly compare $\mathcal{B}$ and $\mathcal{M}$. For $\mathcal{B}/\mathcal{G}$, we have to first expand $\mathcal{B}$ by using every row as many times as the respective name has been annotated with an rID. This is often zero times, which means removing the row. For $\mathcal{M}/\mathcal{G}$, we do the same on $\mathcal{M}$. Comparison $\mathcal{B}/\mathcal{M}$ can be done without any annotation and may be used to estimate expected loss of Recall for a given blocking scheme. Comparison $\mathcal{B}/\mathcal{G}$ is done to get the exact performance if annotation is present. Comparison $\mathcal{M}/\mathcal{G}$ also requires annotation but is independent of the blocking scheme. It allows to get an idea of the relationship between name-matching and actual co-reference. Precision is the number of pairs that are given in $\mathcal{B} \cap \mathcal{M}$, $\mathcal{B} \cap \mathcal{G}$ and $\mathcal{M} \cap \mathcal{G}$ divided by the number of pairs in $\mathcal{B}$, $\mathcal{B}$ and $\mathcal{M}$, respectively. Recall is number of pairs in $\mathcal{B} \cap \mathcal{M}$, $\mathcal{B} \cap \mathcal{G}$ and $\mathcal{M} \cap \mathcal{G}$ divided by number of pairs in $\mathcal{M}$, $\mathcal{G}$

	f2		*f3*		*f4*		*inits*		*e0*		*e5*		*e7*		*e8*		
sizes	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**#**
1-10	**100**	**99**	**100**	87	**100**	89	**100**	96	**100**	94	**100**	95	**100**	96	99	97	58532
11-25	99	**98**	**100**	66	**100**	72	99	92	99	82	99	85	99	91	97	95	23074
26-50	99	**98**	**100**	62	**100**	69	99	91	99	77	99	84	98	91	95	96	16589
51-100	98	**98**	**99**	61	**99**	70	**99**	91	**99**	74	**99**	83	97	92	89	96	13018
101-250	95	**98**	98	61	**99**	71	97	91	97	71	97	82	93	93	79	97	9485
251-500	89	**99**	96	62	**97**	74	94	91	94	69	93	81	86	92	61	97	2833
501-1000	76	**99**	93	62	**94**	76	89	91	89	67	86	77	77	90	42	97	1040

Table 3: Results for different problem size ranges.

and $\mathcal{G}$, respectively. This amounts to the *pairwise-F1* metric [15]. Table 1 gives an intuitive explanation of the meaning of Precision and Recall in the different comparisons.

6.3 Experiments

For our experiments, we normalize all names in the Web of Science into our name representation. For every blocking scheme S, we then iterate over all distinct surnames $l \in L$ for which there is at least one mention with an annotated rID and calculate Precision and Recall for comparisons $\mathcal{B}/\mathcal{M}$, $\mathcal{B}/\mathcal{G}$ and $\mathcal{M}/\mathcal{G}$. This leaves us with one row of results for every pair in $S \times L$. As stated above, comparison $\mathcal{M}/\mathcal{G}$ is indifferent to S. We also store the number of rIDs annotated with the current surname to keep track of the problem size. Separating problem sizes prevents frequent easy cases from dominating the average results and is simply more informative. Finally, for every scheme S and every comparison, we plot the respective Precision and Recall value (one point per surname) against the problem size. A moving average over these points allows for clear analysis of how the performance of a blocking scheme relates to different problem sizes. Comparing these figures for different schemes gives an idea of a scheme's usefulness. Furthermore, for each problem size γ, we plot the average squared size of the largest block over all names $\mathcal{N}_\gamma$:

$$complexity(\gamma) = \frac{\sum_{N \in \mathcal{N}_\gamma} \max_B(|B|)^2}{|\mathcal{N}_\gamma|}$$

This number allows us to estimate a scheme's complexity.

6.4 Results

Results of our experiments are displayed in Figures 7 ($\mathcal{B}/\mathcal{G}$ for 'field-isolate' schemes), 8 ($\mathcal{B}/\mathcal{G}$ for 'entropy-isolate'), 9 ($\mathcal{B}/\mathcal{M}$ for 'field-isolate'), 10 ($\mathcal{B}/\mathcal{M}$ for 'entropy-isolate') and 11 ($\mathcal{M}/\mathcal{G}$). We also report most of the numbers in Table 3. To include surnames with more than 1000 authors, we show results for a selection of the most popular surnames in Table 4. Generally, we view problem size of a surname as number of annotated rID's.

Obviously, in Figures 7 and 8, Precision and Recall are perfect for surnames with just one rID. In general, the larger the blocks are, the worse are results. This is expected as the problem gets more difficult and pairwise evaluation leads to small mistakes having greater consequences in larger blocks (more connections can be missed and this number is squared). There is the usual trade-off in Precision vs. Recall, so that blocking schemes that perform well in one of the two measures are generally among the worse ones in the other. For example in Figure 7, *f2* has highest Recall and lowest

		f3		*f4*		*inits*		*e0*		*e5*		*e7*	
name	**#rID**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Wang	57k	32	56	**35**	65	4	**80**	9	57	7	66	na*	
Lee	35k	**47**	55	38	72	13	**77**	41	55	24	55	na*	
Kim	32k	**41**	54	32	69	10	**76**	15	54	11	56	na*	
Smith	13k	69	56	**76**	91	**76**	**95**	67	55	61	56	22	75
Santos	9k	64	58	**66**	81	57	**85**	62	56	50	69	20	**85**

Table 4: Results for some of the most frequent surnames. *not available due to excessive memory consumption

Precision. However, as explained earlier, for blocking, we have to focus mostly on Recall and computational complexity.

In Figure 7, we see that among blocking schemes of type 'field-isolate', isolating all names without full first name (*f3*) gives worst results in Recall. Isolating all names without second initial (*f4*) is better in terms of Recall, but also not satisfying. Generalizing all names to all available initials (*inits*) gives a Recall of about 90% and, not surprisingly, generalizing all names to surname and first initial by isolating all names with only surname (*f2*) has almost perfect Recall when compared against the gold annotation. This scheme has relatively low Precision, while the other 'field-isolate' schemes all have a very good average Precision of over 90%. The computational complexity of a surname with one thousand different annotated authors exceeds one million in terms of squared size of the largest created block for *f2* but is very moderate for the other schemes of the 'field-isolate' type. In Figure 8, we see that among blocking schemes of type 'entropy-isolate', Recall generally increases and Precision decreases with the threshold deployed in the entropy criterion. The lower the threshold, the more blocks are created. From this follows that the complexity increases with the threshold, although *e0*, *e5* and *e7* are almost indistinguishable in this respect. At first sight, it can be seen that Precision decreases much faster for these schemes than for 'field-isolate', while Recall values are more or less comparable. The complexity of 'entropy-isolate' schemes is considerably higher than for 'field-isolate'.

Comparing blocking against name-matching in $\mathcal{B}/\mathcal{M}$ (Figures 9 and 10), we note that results are comparable to $\mathcal{B}/\mathcal{G}$ (Figures 7 and 8). The order of performance is the same in both Recall and Precision and Precision values are almost identical. However, Recall values differ more and are generally lower. This is not surprising as perfect Recall in $\mathcal{B}/\mathcal{M}$ is only possible if a lot of Precision is sacrified. Exactly this point is displayed clearly in $\mathcal{M}/\mathcal{G}$ (Figure 11), where we can see that the transitive closure of name-matching necessarily connects a number of different authors that is increasing with the popularity of a surname.

6.5 Discussion

Analyzing the blocking schemes to see which ones have the best properties, we remember that Recall is more important than Precision. With that in mind, we can already eliminate schemes *f3*, *f4*, *e0* and *e5* as we cannot accept about 20% mistakes (see Figures 7 and 8). Although these schemes are not as effective, it is interesting to see Recall *increasing* for *f3* and *f4* at problem sizes larger than 100 different annotated authors. This can be explained by authors with popular names being aware of the redundancy of their names and therefore using more complete versions with full first name (*f3*)

and/or second initial (*f4*). Precision is usually very good, but for *e8* it is much worse than for *f2*, although the latter has even better Recall. This and its bad complexity renders *e8* useless. Having a closer look at complexity, we note that it grows approximately linearly for all schemes except *f2*, which already takes around one million comparisons for surnames with around one thousand authors, growing fast. Therefore, *f2* can be eliminated, too. Although complexity of the entropy schemes is generally much higher than for the field schemes (except *f2*), with around 100,000 comparisons for 1000 distinct authors, it appears acceptable for *e7*. We are left with inits and *e7*. The latter has slightly better Recall but inits has much better Precision. Ultimately, *inits* looks like the superior blocking scheme, as it is also much cheaper complexity-wise (it creates smaller blocks) and Table 4 shows that it is clearly better than e7 for some of the most frequent surnames. Still, *e7* constitutes a serious alternative.

Looking at $\mathcal{B}/\mathcal{M}$ (Figures 9 and 10), we see that *inits* has less mismatches in each block (because blocks are smaller), but it also misses much more matches than *e7*. As annotation-based Recall is hardly any worse for *inits* than for *e7*, either (a) matches missed by *inits* are mostly irrelevant as they do not correspond to different references to the same autor, or (b) they are relevant, but the *annotation is such that these mismatches are not penalized* (i.e., because authors that annotate their rID also normalize their names (or this is done automatically afterwards)). This would be in line with observations that often, there is just one or two annotated rIDs per surname, and each is given many times to mentions with the exact same name. ORCID author identifiers should be comparable to rID's (see [13]). For these, Kim [11] has shown that performance of the *inits* method compared to more complex approaches is dependent on the data selection, in particular the number of distinct authors. This could justify *e7* and the idea of entropy-isolate in general. Another rather unexpected observation in $\mathcal{B}/\mathcal{M}$ is that the surname, first initial scheme (*f2*) has far from perfect Recall, which suggests that there is a good number of cases where only the last name is given – or different initials are used for the same person. A likely explanation is that when comparing to the MATCH relation, an isolated surname accounts for a large number of missed matches (pairings with all observed names below it). This can also explain the low Recall values for *inits* in $\mathcal{B}/\mathcal{M}$.

7 CONCLUSION

RQ1: How to structure names, their variations and relations? We parse all names in the Web of Science into a format that allows to specify a surname and up to three first names fully or by initial. Specification in terms of adding or completing initials can be encoded in a PARENT relation defining a directed acyclic graph in which a name's children are its direct specifications.

RQ2: How to include name-matching in the blocking? Two names match, if they are not contradictory. The MATCH relation is a subset of the BLOCK relation – the symmetric transitive closure of the PARENT relation with two types of hypothetical (unobserved) nodes added. Name-matching is not transitive, but as blocks partition the set of author mentions, we need to end up with equivalence classes defined by equivalence relations, which by definition have to be reflexive, symmetric – and transitive. Therefore,

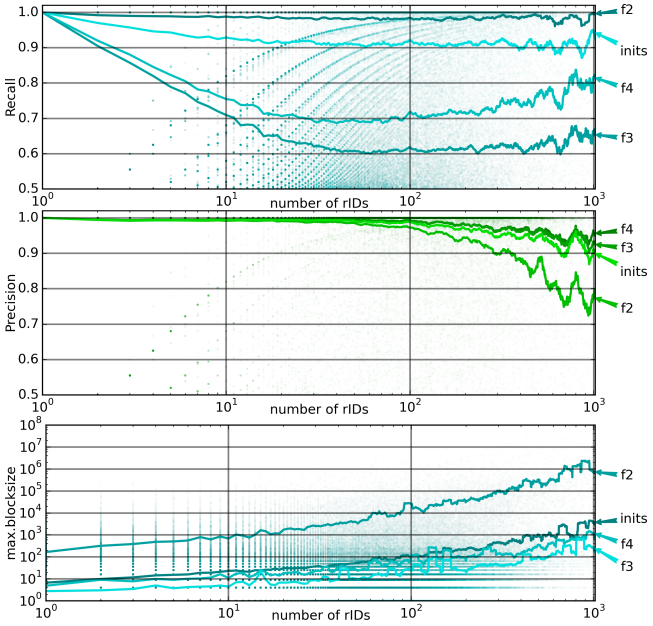

Figure 7: B/G: Recall, Precision and complexity for blocking of type 'field-isolate' compared to GOLD annotation.

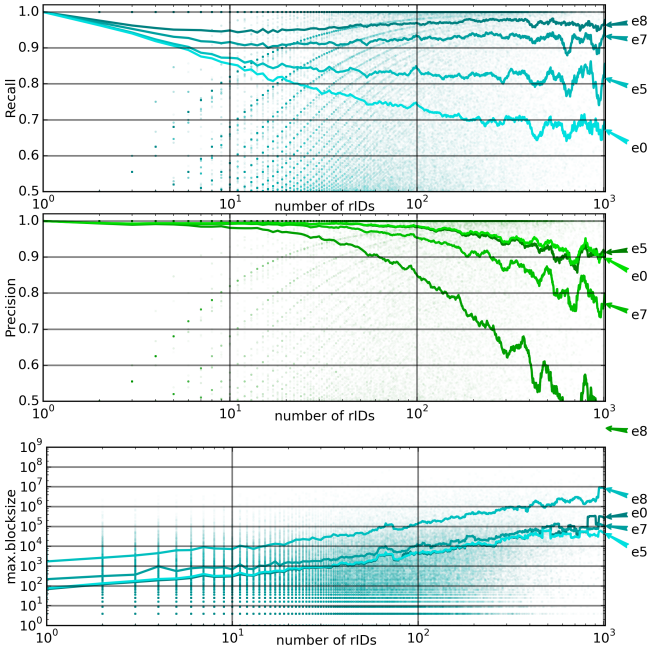

Figure 8: B/G: Recall, Precision and complexity for blocking of type 'entropy-isolate' compared to MATCH relation.

we transform name-matching into blocking through the PARENT relation and the transitive closure operation. We define blocking schemes as sets of names to isolate in the graph, before generating blocks using the closure. Traditional blocking schemes can be encoded in this framework.

RQ3: How do different blocking schemes perform? The BLOCK relation can be compared to the GOLD equivalence relation defined

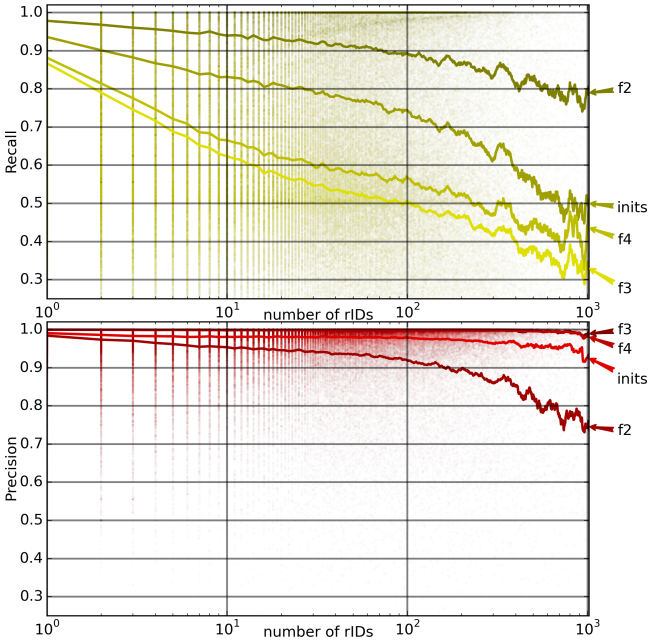

Figure 9: $\mathcal{B}/\mathcal{M}$: Recall, Precision and complexity for blocking of type 'field-isolate' compared to GOLD annotation.

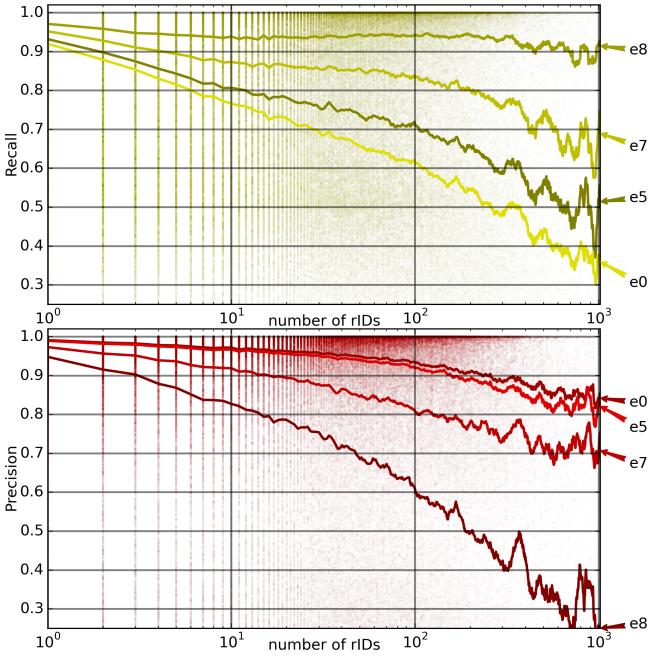

Figure 10: $\mathcal{B}/\mathcal{M}$: Recall, Precision and complexity for blocking of type 'entropy-isolate' compared to MATCH relation.

by annotated rIDs in the Web of Science. It can also be compared to the MATCH relation. Among traditional blocking schemes, generalizing all first names to their initials (*inits*) is clearly the best choice, although this separates mentions of the same author when a different number of initials is specified. Generalizing all names to surname, first initial (*f2*) is too expensive for frequent names.

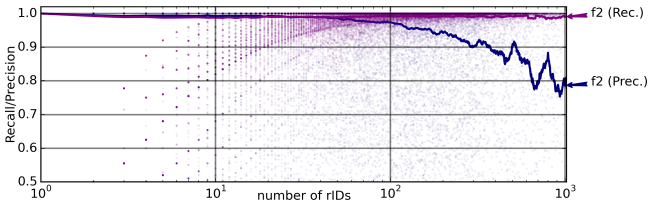

Figure 11: $\mathcal{M}/\mathcal{G}$: Recall and Precision for MATCH evaluated against GOLD. The blocking scheme is irrelevant.

Isolating all names without full first name (*f3*) has surprisingly low Recall. Unexpectedly, isolating all names without second initial (*f4*) has better Recall than *f3*, although not all people even have a second name. Performance of different traditional schemes is strongly influenced by conventions about which parts of a name are specified. This varies by culture (i.e., in China, first names are more specific) and by surname frequency. In a human effort to reduce entropy, frequent surnames are generally reported with more first name information.

RQ4: Are there better blocking schemes? As ultimately, the purpose of names is to reduce entropy in a set of people, we have tested a straightforward blocking scheme in which names are isolated if their connecting more specific nodes introduces too much entropy in terms of name-matching. Different values were tested for the threshold and the best was 0.75. This scheme constitutes an alternative to the *inits* scheme. However, the latter still performs better in our evaluation. In future work, it shall be determined whether this is because it better models different authors or it better models the *annotation patterns* in the WoS. Another task for the future is to exploit the conceptional benefits of our framework (in which a blocking scheme is nothing but a set S of names to isolate) and evaluate schemes that treat surnames differently depending on their popularity or culture, perhaps even considering specific names or groups of names for isolation.

RQ5: Can name-matching substitute author annotation? The MATCH relation can be compared against the GOLD relation to see how name-matching and author identification (or at least author annotation) correlate. Also, one can compare the results $\mathcal{B}/\mathcal{G}$ (Figures 7 and 8) of different blocking schemes given the GOLD relation against the results $\mathcal{B}/\mathcal{M}$ (Figures 9 and 10) of these schemes given the MATCH relation. A general limitation is that MATCH – in contrast to GOLD – is not an equivalence relation. For this reason, we see in $\mathcal{B}/\mathcal{G}$ that the MATCH relation includes almost all pairs that need to be found (close to perfect Recall) but introduces an expected growing number of mismatches as all contradictory matches are included (i.e., *J. Doe* is *John Doe* and *Jack* Doe at the same time). However, based on the indirect comparison $\mathcal{B}/\mathcal{G}$ vs. $\mathcal{B}/\mathcal{M}$, we can say that name-matching is an acceptable way for preliminary evaluation of blocking schemes if author annotation is not available: results of $\mathcal{B}/\mathcal{M}$ suffice to determine the order of results in $\mathcal{B}/\mathcal{G}$. Together with observing the complexity of different schemes, this should be enough to select the best of a number of schemes compared. In the AND setting, blocking inevitably combines names that do not match. The phenomenon is quantified in $\mathcal{B}/\mathcal{M}$. $\mathcal{M}/\mathcal{G}$ (Figure 11) shows that names referring to the same author almost always match (Recall in $\mathcal{B}/\mathcal{M}$). Naturally, not

all matching names refer to the same author (Precision in $\mathcal{B}/\mathcal{M}$). Generally speaking, an AND method should not group mentions of authors whose names do not match. Therefore, we can use the MATCH-relation to obtain a Precision value for AND methods, as suggested by Schulz et al. [20]. Unfortunately, there is a tradeoff with Recall, which is not available in this setup (separated matching names are not necessarily mistakes), so we cannot use MATCH as a proxy for optimizing AND.

RQ6: What impact has blocking on (the evaluation of) AND?. In the blocking and disambiguation setup, there are a number of different options to obtain Precision and Recall values. For blocking, there is no choice: Precision values are computed for each block and may then be aggregated (weighted or separated by problem size). We have to evaluate bRec with reference to a larger context, here the entire collection. Aggregating disambiguation Precision (dRec), it is important to weight the values by squared cluster/block size – or to report them separately for each size. Otherwise, easy problems will dominate the average due to the Zipfian distribution of block sizes [15, 20]. Precision of a larger context (i.e., the current block) is equivalent to the weighted average of the smaller components' Precision (i.e all authors in this block). Disambiguation Recall (dRec) is most critical: If computed for each block and then aggregated, it misses connections across blocks. Therefore, Recall should always be computed for the entire collection. If Precision and Recall are computed as recommended above, bPrec sets a lower bound on dPrec and bRec sets an upper bound on dRec. This is the impact of blocking on author disambiguation. A combination of Precision and Recall in the F-measure only makes sense if both are determined over the same set. Both blocking and disambiguation are evaluated based on gold author identifiers. When it comes to AND, one might think that initially, Precision is zero and Recall is one. However, the contrary is true: Our evaluation has shown that usually, bPrec is already close to one and bRec is considerably lower than one as blocking always misses a number of author pairs. Still, many AND papers do not use their blocking as a baseline. Perhaps it would be best to report *gain* in Precision and *loss* in Recall in a pre-post comparison. The case of high bPrec holds in particular for the frequent case of less popular names. These have much higher bPrec than popular names (Recall is only slightly lower for popular names). Clearly, AND is particularly useful for popular names and methods should be developed focusing on such names. We stress once more that there are two potential confounders for high Precision of AND: (a) High bPrec (data and blocking), (b) unweighted average over a Zipfian distribution of problem sizes (data and evaluation). Furthermore, there is a danger of confusion over the actual number of mistakes when average Recall values are reported. Finally, author names contain some of the most specific information to discriminate different authors. Due to their transitive nature, blocks alone do not do justice to the fact that this information is best suited for pairwise comparisons. Therefore, besides blocking, we also recommend using names as features in AND.

ACKNOWLEDGMENTS

This work was supported by the EU's Horizon 2020 programme under grant agreement H2020-693092, the MOVING project.

REFERENCES

[1] Peter Christen. 2012. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering* 24, 9 (2012), 1537–1555.
[2] Ian Davidson and SS Ravi. 2005. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 138–149.
[3] Ana Paula De Carvalho, Anderson A Ferreira, Alberto HF Laender, and Marcos A Gonçalves. 2011. Incremental unsupervised name disambiguation in cleaned digital libraries. *Journal of Information and Data Management* 2, 3 (2011), 289.
[4] Anderson A Ferreira, Marcos André Gonçalves, and Alberto HF Laender. 2012. A brief survey of automatic methods for author name disambiguation. *Acm Sigmod Record* 41, 2 (2012), 15–26.
[5] Carmen Galvez and Félix Moya-Anegón. 2007. Approximate personal name-matching through finite-state graphs. *Journal of the Association for Information Science and Technology* 58, 13 (2007), 1960–1976.
[6] Thomas Gurney, Edwin Horlings, and Peter Van Den Besselaar. 2012. Author disambiguation using multi-aspect similarity indicators. *Scientometrics* 91, 2 (2012).
[7] Anne-Wil Harzing. 2015. Health warning: might contain multiple personalitiesthe problem of homonyms in Thomson Reuters Essential Science Indicators. *Scientometrics* 105, 3 (2015), 2259–2270.
[8] Ijaz Hussain and Sohail Asghar. 2017. A survey of author name disambiguation techniques: 2010–2016. *The Knowledge Engineering Review* 32 (2017).
[9] Yannis E Ioannidis, Raghu Ramakrishnan, et al. 1988. Efficient Transitive Closure Algorithms.. In *VLDB*, Vol. 88. 382–394.
[10] Hakan Kardes, Deepak Konidena, Siddharth Agrawal, Micah Huff, and Ang Sun. 2013. Graph-based approaches for organization entity resolution in mapreduce. *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing* (2013), 70–78.
[11] Jinseok Kim. 2018. Evaluating author name disambiguation for digital libraries: a case of DBLP. *Scientometrics* (2018), 1–20.
[12] Kunho Kim, Athar Sefid, and C Lee Giles. 2017. Scaling Author Name Disambiguation with CNF Blocking. *arXiv preprint arXiv:1709.09657* (2017).
[13] Thomas Krämer, Fakhri Momeni, and Philipp Mayr. 2017. Coverage of Author Identifiers in Web of Science and Scopus. *arXiv preprint arXiv:1703.01319* (2017).
[14] Biji T Kurien. 2008. Name variations can hit citation rankings. *Nature* 453, 7194 (2008), 450.
[15] Michael Levin, Stefan Krawczyk, Steven Bethard, and Dan Jurafsky. 2012. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the Association for Information Science and Technology* 63, 5 (2012), 1030–1047.
[16] Staša Milojević. 2013. Accuracy of simple, initials-based methods for author name disambiguation. *Journal of Informetrics* 7, 4 (2013), 767–773.
[17] Natsuo Onodera, Mariko Iwasawa, Nobuyuki Midorikawa, Fuyuki Yoshikane, Kou Amano, Yutaka Ootani, Tadashi Kodama, Yasuhiko Kiyama, Hiroyuki Tsunoda, and Shizuka Yamazaki. 2011. A method for eliminating articles by homonymous authors from the large number of articles retrieved by author search. *Journal of the American Society for Information Science and Technology* 62, 4 (2011), 677–690.
[18] Gerald Penn. 2006. Efficient transitive closure of sparse matrices over closed semirings. *Theoretical Computer Science* 354, 1 (2006), 72–81.
[19] Jane Qiu. 2008. Scientific publishing: identity crisis. *Nature News* 451, 7180 (2008), 766–767.
[20] Christian Schulz, Amin Mazloumian, Alexander M Petersen, Orion Penner, and Dirk Helbing. 2014. Exploiting citation networks for large-scale author name disambiguation. *EPJ Data Science* 3, 1 (2014), 11.
[21] Andreas Strotmann and Dangzhi Zhao. 2012. Author name disambiguation: What difference does it make in author-based citation analysis? *Journal of the American Society for Information Science and Technology* 63, 9 (2012), 1820–1833.
[22] Jie Tang, Alvis CM Fong, Bo Wang, and Jing Zhang. 2012. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering* 24, 6 (2012), 975–987.
[23] Li Tang and John Walsh. 2010. Bibliometric fingerprints: name disambiguation based on approximate structure equivalence of cognitive maps. *Scientometrics* 84, 3 (2010), 763–784.
[24] Vetle I Torvik and Neil R Smalheiser. 2009. Author name disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3, 3 (2009), 11.
[25] Xuezhi Wang, Jie Tang, Hong Cheng, and S Yu Philip. 2011. Adana: Active name disambiguation. In *2011 11th IEEE International Conference on Data Mining*. IEEE.